Paper

Theoretical and Experimental Analysis of Cryptographic Hash Functions

Jacek Tchórzewski[1,2] and Agnieszka Jakóbik[2]

[1] *AGH University of Science and Technology, Cracow, Poland*
[2] *Cracow University of Technology, Cracow, Poland*

https://doi.org/10.26636/jtit.2019.128018

Abstract—The paper presents a theoretical introduction to the cryptographic hash function theory and a statistical experimental analysis of selected hash functions. The definition of hash functions, differences between them, their strengths and weaknesses are explained as well. Different hash function types, classes and parameters are described. The features of hash functions are analyzed by performing statistical analysis. Experimental analysis is performed for three certified hash functions: SHA1-160, SHA2-512 and SHA3-512. Such an analysis helps understand the behavior of cryptographic hash functions and may be very helpful for comparing the security level of the hashing method selected. The tests may serve as a basis for examination of each newly proposed hash function. Additionally, the analysis may be harness as a method for comparing future proposals with the existing functions.

Keywords—cryptographic hash function, hashing metod, security.

1. Introduction

As they play an important role in ensuring the security and confidentiality of information, identification and authentication methods are approached with an ever greater attention, both in civilian (personal information, passwords, PIN codes) and military domains. Hashing is one of the techniques enabling to meet some of the demands described above. Practical applications of cryptographic hash functions include message integrity checking, digital signatures, authentication procedures and other information security-related applications.

The paper is organized as follows. In Section 2 we describe the properties of one way functions, as well as the properties and classes of hash functions. In Section 3, methods of creating hashing functions are presented. In Section 4, the strengths and weaknesses of hashing functions certificated by NIST are presented. Section 5 is devoted to statistical tests involving SHA1-160, SHA-512 and SHA3-512, with their results described. Section 6 summarizes the work and offers conclusions.

2. Hash Functions: Properties, Classes and Types

Let us start with the definition of a one way function, which is given below [1]:

$$\forall x \in X, f : x \rightarrow y \wedge \neg(\exists g : y \rightarrow x) \ . \tag{1}$$

It means that for all function arguments x there exist a value y, but it is impossible to identify a function which will assume this value y as an argument and return x. Hash functions belong to family of one way functions, but are bound by an additional restriction. Formally, they are defined as follows [2]:

$$h : \{0,1\}^* \rightarrow \{0,1\}^n, n \geq 1 \ , \tag{2}$$

where $\{0,1\}^*$ is an input set (formally its elements may be of any length), and will be further denoted by M. Elements from M will be denoted by m ($m \in M$). $\{0,1\}^n$ is an set of output hashes, each with a fixed length and a finite number of combinations, and will be further denoted by H (note, that n is greater than or equal to 1) [2]. Hashes from H will be denoted by $h(m)$ ($h(m) \in H$).

In article [3] Carter and Wegman presented three basic hash function classes.

2.1. Universal Hash Functions Classes

Class H1 is designated for computers which are capable of fast multiplication of the input bit string. Hashes from this class may become inconvenient when the input bit string is too long to multiply it in a single machine instruction. The basic formula of hashes from this group is: for 2 elements, let us call them m and n, the hash is calculated as follows [3]:

$$h_{m,n}(x) = (mx+n) \mod p \ . \tag{3}$$

In class H3, only simple linear transformations are used instead of multiplication. Formally, the class is defined as follows: if the hash function transforms elements from

set A (each element is a binary number with length i) to set B (each element is a binary number with length j), M is an array of size i and contains elements from B, and m are elements of M ($m \in M$) where $m(k)$ is the k-th bit of element m, then for any $x \in A$ (with the same bit indexing as m), the hash function is represented by [3]:

$$h_m(x) = x_1 m(1) \oplus x_2 m(2) \oplus \cdots \oplus x_i m(i). \quad (4)$$

Then H3 is a set defined in the following manner [3]:

$$\{f_m : m \in M\} \ . \quad (5)$$

Class H2 is very similar to class H3. The difference is that hashing functions from this class require more space for hash computation, but need less time. The key point is to find a function g which maps an input bit string into a longer input bit stream containing fewer less '1s'. Then, H2 can be defined [3]:

$$\{f * g : f \in H3\} \ . \quad (6)$$

To define when a set of hashing functions becomes universal, we have to introduce a certain notation. Let us consider hash function h which maps set A into set B. It is always assumed that $|A| > |B|$. Then, it is possible to define function δ_h in the following manner [3]:

$$\delta_h(x,y) = \begin{cases} 1 & \text{if } x \neq y \text{ and } h(x) = h(y) \\ 0 & \text{otherwise} \end{cases} \ , \quad (7)$$

where $x, y \in A$. We can say that collection of hashing functions C is universal when for all x and y in A $\delta_C(x,y) \leq \frac{|C|}{|B|}$ [3]. In practice, this means that no pair of distinct inputs from A collides under more than $(\frac{1}{|B|})-$th of the functions [3]. All three classes (H1, H2, and H3) are universal, but H2 and H3 classes are the most popular ones [3].

2.2. Hash Function Types

Cryptographic hash functions may be divided into two groups [2]:

- keyed hash functions – require a secret key and are known as message authentication code (MAC) [2],
- un-keyed hash functions – do not require any secret key and may be referred to as manipulation detection code (MDC).

  Generally, the term hash functions refer to un-keyed hash functions [2].

In this paper, we will focus on un-keyed hash functions which can be divided into three subgroups, based on their additional properties:

1. One way hash functions (OWHF) – defined by Merkle [4] and fulfilling the following requirements:
   - hash function does not give any constraint on input data size,
   - output hash has constant length,
   - output hash should be easy to compute,
   - "given h and $h(x)$, it is computationally infeasible to determine x" – a preimage resistance feature,
   - "given h and x, it is computationally infeasible to find an $x' \neq x$ such feature that $h(x) = h(x')$" – the second preimage resistance.
2. Collision resistant hash functions (CRHF) – belonging to the OWHF group and fulfilling an additional requirement: it is impossible to find a pair (x,x') where $x \neq x'$, which have the same hash value $(h(x) = h(x'))$. This condition is known as collision resistance. The difference between the second preimage resistance depends on the selection of arguments. In the second preimage resistance condition, the attacker has a given value x and has to find x'. In the collision resistance condition, the selection of both: x and x' is a free choice of the attacker.
3. Universal one way hash functions – a family in which the probability of finding a second preimage for a randomly chosen hash function is negligible [2], [5]. These functions are faster than CRHF and allow to omit trapdoors during digital signature creation. They are used when it is impossible to make a decision in which the hash function should be chosen before computation starts.

3. MDC Construction Method

3.1. Hash Function Based on Block Ciphers

To describe the general concept of creation of hash functions based on block ciphers, the following set have to be defined [6], [7]:

$$S \in \{M_i, M_i \text{ XOR } X_i, X_i, C\} \ , \quad (8)$$

where M_i is one block of a message, X_i is a chaining value from the previous step and C is a chosen constant value [7]. Note that all these values are given for i-th round of hash computation, and that secure block cipher B was already chosen. Then the construction of i-th round is:

1. Choose a private key P for B from set S.
2. Choose an input I from set S.
3. Pass I and P to the algorithm B and calculate cipher value CV.
4. Choose value T from set S.
5. Calculate $X_{i+1} = T \oplus CV$.
6. Update set S with values X_{i+1} and M_{i+1} (next block of message) according to the formula (8). If it is impossible, computation ends.
7. Go to step 1.

The selection of variables depends on the algorithm design, but at least one variable should be M_i. The output hash should be as big as block size of B, or twice as big [6]. This is caused by the small size (mainly 64 bits) of the block. Hash should be bigger to avoid collisions. The speed of hash functions based on block ciphers is equal to the number of encryptions to process r plaintext bits, where r is defined as block size [6].
Most hash functions constructed in this way suffer from numerous security vulnerabilities [7] and cannot be used in practice. However, an opposite situation may occur when the block cipher construction is based on the hash function, for example in SHACAL and SHACAL-2, with both being based on the SHA-1 cryptographic hash function [2]. A good example of a hash function with its length equal to the size of the block is described by Meyer and Oseas in [8]. More examples may be found in [6].

3.2. Hash Functions Based on Cellular Automata

Cellular automata (CA) can be used for ciphers generations and for hash functions design [7], since Wolfram [9] developed a pseudorandom generator based on CA rule 30. Cellhas, as described in [10], is a good example of a hashing function based on CA.

3.3. Hash Functions Based on Math

There are three ways of creating of hashing functions based on mathematical constructions:

1. Hashing function based on mathematical primitives is based on modular arithmetic, discrete logarithm problem and factorization problem [11].
   - Factorization problem is based on the difficulty of finding two factors, for any positive integer, which, when multiplied, will give these integers. This problem can be also described by the following formula: for a given integer I it is hard to find a and b such that $ab = I$.
   - Discrete logarithm problem, such as that for a given abelian group O, generator of this group o and an element e which belongs to O, finding (if it is possible) x such that $o^x = e$. The difficulty of the discrete logarithm problem depends on group O.
2. Hashing function based on Knapsack NP-complete problem. From a cryptographic point of view its formula may be formulated as [6]:

$$\sum_{i=1}^{n} a_i \cdot x_i = S \ , \tag{9}$$

   where each a_i is a m bit integers: $\{a_1, a_2, \ldots, a_n\}$, S is a p bit integer and $p \approx m + \log_2 n$, and X is a vector of elements $x_i \in \{0,1\}$, [6], [7] and [11].
3. Hashing function based on algebraic matrices, developed by Harari [12]. Here, the key K is a $n \times n$ random matrix and M is the $1 \times n$ message matrix. Then the digest D is: $M^T KM$ or equivalently $K^T MK$ [7], [12]:

$$D = M^T KM. \tag{10}$$

   or

$$D = K^T MK. \tag{11}$$

   Unfortunately, collisions appeared in the Harari hash function proposition [7].

3.4. Dedicated Hash Functions

Dedicated hash functions created only for hashing operation. Their security can be proved mainly in an empirical way, because they very often do not base on any hard problem, like factorization or discrete logarithm problem. Examples are MD4, MD5, SHA1, SHA2 or SHA3. Based on them algorithms were designed to be as fast as possible in software implementations rather than hardware [11].

3.5. Standardization of Hashing Functions

After the first collision for MD5 was discovered, the National Institute of Standard and Technology, USA, created a hashing standard – Secure Hash Algorithm (SHA). The first version of SHA, known as SHA-0, was published in 1993. In 1995 SHA-0 was replaced by a new version – SHA-1. In 2005 vulnerabilities were identified in SHA-1 and NIST introduced SHA-2, which is used currently. In 2007 an open competition for the next generation SHA-3 was announced. The evaluation criteria are as follows [13]:

- applications of the hash functions – the wider variety of cryptographic usage, the better. The new standard should be useful for the creation of hashed message authentication code (HMAC), as well as for the creation of digital signatures or random bit generators [14].
- specific requirements when hash functions are used to support HMAC, pseudo-random functions (PRFs), or randomized hashing – each algorithm had to have at least one scheme to support HMAC as PRFs [14]. These PRFs have to be secure against known attacks which require less than $2^{\frac{n}{2}}$ queries or which require less computation then the preimage attack [14]. If the hashing function is capable of randomized hashing, it has to have n security bits against attacks mentioned in [14].
- additional security requirements of hash functions – for a digest with size n: $\frac{n}{2}$ bits of collision resistance, n bits of preimage resistance and $n-k$ bits second preimage resistance for any message shorter than 2^k bits [14]. All these rules should be fulfilled with m replacing n for any m size subset taken from the digest [14].

- evaluation of attack resistance- hashing functions were attacked with well-known and popular methods discovered during the security evaluation phase. Other validation methods were based on statistical and behavioral tests, such randomness of hash creation [14].
- other consideration factors – for example quality of security proofs, proper analysis, documentation and simplicity of the algorithm, as well as opinions by NIST and the cryptographic community.

The remaining criteria included speed of the algorithm, code size, memory and hardware implementation requirements, flexibility and simplicity [13]. The final report announcing the winner (the Keccak algorithm) was published in 2012 [15].

4. Theoretical Analysis of Security Parameters

In this section, we will present the results of an analysis of the dependence between the length of the digested messages and the security parameters of hashing functions.
The security level of a cryptographic primitive is expressed in bits, where n-bit security means that the attacker would have to perform 2^n operations to break it. The security level of a cryptographic hash function has been defined using the following properties:

- collision resistance bits of security,
- preimage attacks bits of security,
- second preimage resistance bits of security.

To compromise collision resistance using the brute force method, the attacker needs to hash a huge number of variants of the message m, and hash a huge number of variants m', go through the lists and see if there are values that are equal. For example, in a 160-bit hash output, the attacker needs 2^{80} inputs to test in both lists. Therefore, in the case of this hash function, the number of bits of security against this attack is equal to 80, due to the Birthday Paradox.
While breaking preimage and second preimage resistance, the attacker cannot apply the Birthday Paradox. For a mbox160-bit hash output, the attacker needs to examine 2^{160} input messages, which means that 160 bits of security are achieved.
In Table 1, security parameters of selected hashing functions, as accepted by NIST, are presented [16]. In Table 1, function L(M) is defined as:

$$L(M) = \left\lceil \log_2 \frac{\text{len}(M)}{B} \right\rceil . \tag{12}$$

where M is the input message, B is the block size of the hash function and $\lceil . \rceil$ denotes the least integer not strictly lower than the argument in the brackets.

Table 1
NIST-approved security parameters of hash functions

		Bits of security		
Function	Output size	Collision	Pre-image	Second preimage
SHA-1	160	< 80	160	160-L(M)
SHA-224	224	112	224	min [224, 256-L(M)]
SHA-256	256	128	256	256-L(M)
SHA-384	384	192	384	384
SHA-512	512	256	512	512-L(M)
SHA3-224	224	112	224	224
SHA3-256	256	128	256	256
SHA3-384	384	192	384	384
SHA3-512	512	256	512	512

Using the brute force method, there always exists a generic attack comprising $2^{\frac{n}{2}}$, 2^n and 2^n steps, respectively, where n is the hash length [17]. This is the maximum (ideal) security level which can be achieved for any hash function. As it can be seen in Table 1, SHA-1 has a lower-than-ideal security level in terms of collision attacks and second preimage attacks. SHA-1 offers the maximum potential strength in terms of second preimage attacks, when the message size (in bits) is up to 160. With bigger message sizes, Eq. (12) is growing up to 1 (Fig. 1).

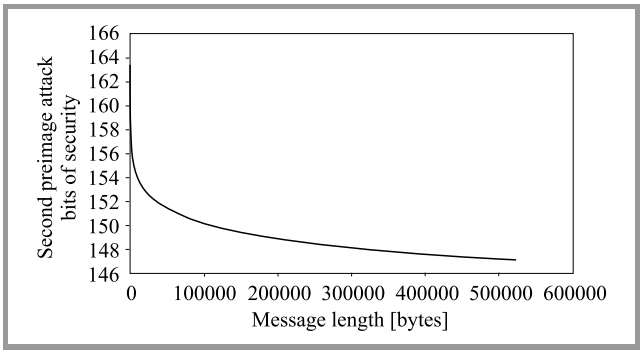

Fig. 1. SHA-160 second preimage attack bits of security.

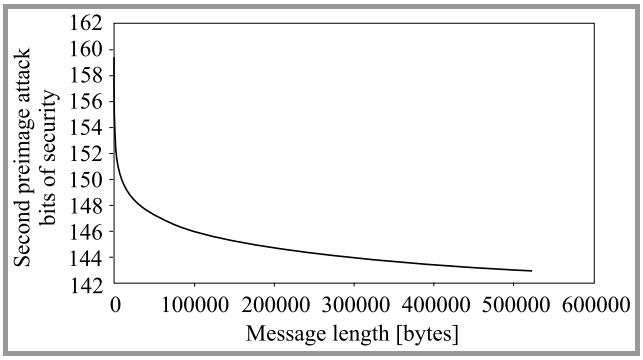

Fig. 2. SHA-256 second preimage attack bits of security.

The SHA-2 family is collision resistant but in every case (except for SHA-384), the security bit parameter of the second preimage attack cannot be ideal when the length

of M is greater than the block size B. Dependencies between the number of bits of security and the message length for SHA-256 and SHA-512 are presented in Figs. 2 and 3. For SHA-256, resistance to second preimage attacks is not perfect when the message size is over 256 bits. the maximum message size was measured in the same way as in the case of SHA-1.

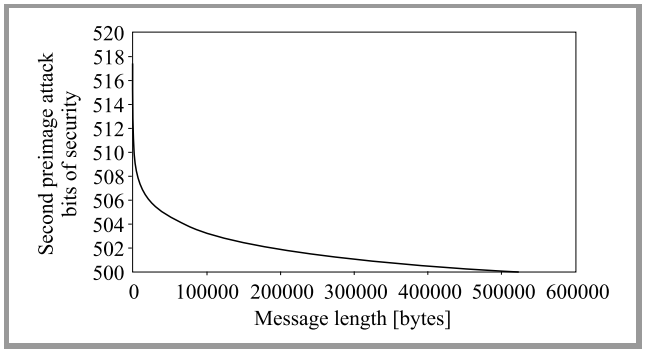

Fig. 3. SHA-512 second preimage attack bits of security.

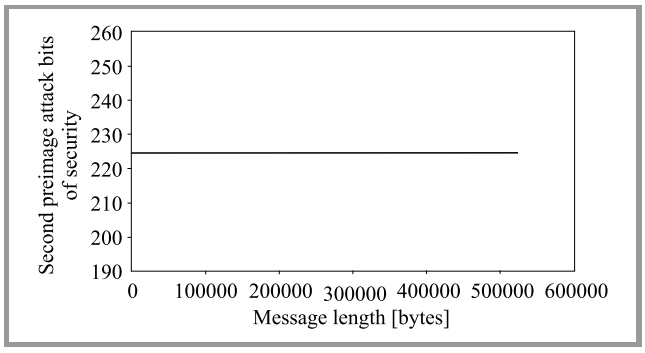

Fig. 4. SHA-224 second preimage attack bits of security.

The case of SHA-512 is similar to that of SHA-256, but this time the maximum secure message size is 512 bits. It is so because Eq. (12) is greater than or equal to 1 only when $\log_2 \frac{\text{len}(M)}{B}$ is greater than 0. $\log_2 \frac{\text{len}(M)}{B}$ is greater than 0 only when the message length is greater than the block size. SHA-224 is different than its predecessors. As can be seen in Fig. 4 the dependence between the number of bits of security and the message size is constant, but according to NIST (Table 1), SHA-224 is not totally resistant against second preimage attacks. This is because the security bits are defined in this case by $\min[224, 256-\text{L}(M)]$. A vulnerability appears when $\text{L}(M) = 33$ $(56-33 = 223)$. This situation may occur when $\left\lceil \frac{\text{len}(M)}{512} \right\rceil = 2^{33}$. Thus, $\frac{\text{len}(M)}{512}$ should be in the range of $(2^{33}-1, 2^{33})$. $\frac{\text{len}(M)}{512} = 2^{33}-1$ when $\text{len}(M) = 4398046510592$ bits. It means that SHA-224 becomes insecure against second preimage attacks when the size of M is about 512 GB.

The SHA3 function family currently offers perfect security against all three attacks: collision, preimage and second preimage.

5. Experimental Analysis for Selected NIST Hash Functions

For statistical analysis, we have chosen the strongest (the longest) hash function from the SHA1, SHA2 and SHA3 families, i.e. SHA-512, SHA3-512 and SHA1-160, respectively. A data sample consisting of 10,000 messages was considered. All tests were implemented in the JAVA programming language (JDK 1.8) and hashes were generated with the use of the Bouncy Castle library [18].

All 10000 random messages were binary strings. Each input had the same length as the output digest size (160 bit inputs for SHA1 function and 512 bit inputs for SHA-512 and SHA3-512 functions). All input data was generated one by one, separately for SHA1, SHA-512 and SHA3-512, with the use of SecureRandom Java class [19].

By hashing those inputs, we have received the same number pairs: $(input, digest)$ for every hash function.

Three statistical tests were performed: hamming distance test, bits probability test and series test. The details, results, conclusions and comparisons are described in the following subsections.

5.1. Hamming Distance Test

The idea of this test was to measure how small (or even micro) changes in input data influence the output hash. Hash function is passing the test when the statistic $|\text{Z}|$ from T-Student test (13) is within the $(0, 1.96)$ interval. The expected value is equal to $\frac{Hashsize}{2}$ (50%). Significance level α was set to 5%. The T-Student formula is:

$$|Z| = \left| \frac{Average\,value - Expected\,value}{Standard\,deviation} \sqrt{Sample\,size} \right|. \quad (13)$$

Firstly, we generated, for each of the pairs $(input, hash)$, another pair $(input', hash')$, where $input'$ was the original input with one random bit changed to the opposite (1 into 0 or 0 into 1), and $hash'$ was a digest calculated from input$'$. We have received two very similar inputs and two hashes. The aim of the experiment was to measure the Hamming distance [20] between these hashes and to repeat this procedure for all inputs generated and for all hashing functions chosen. The hamming distance may be defined as follows. If $S1$ is the first bit string, $S2$ is the second bit string and $\text{len}(S1) = \text{len}(S2)$, the Hamming distance between $S1$ and $S2$ is the number of 1s in the string $S3 = S1 \oplus S2$. It is the number of positions in which $S1$ have different values than $S2$, which can be interpreted as the distance between $S1$ and $S2$.

Results of the experiment for the SHA1 hashing function are presented in Fig. 5 and in Table 2. The horizontal black line is set to 80 because it is the expected value (distance). 4673 out of 10, 000 values were over the black line, but the score is close to 50%. The critical values presented in Table 2 indicate that the average is almost 50%. The $|Z|$ statistic was equal to 1.17, so the T-Student test had been passed. The conclusion is that micro changes in input data

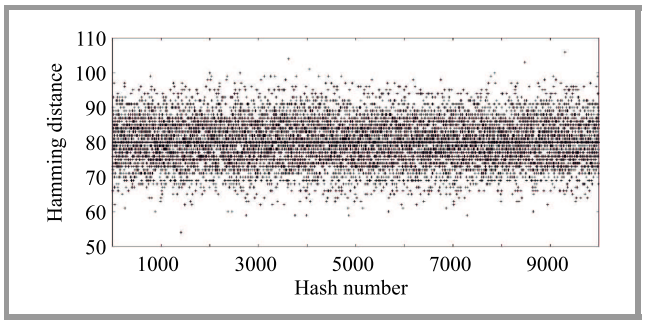

Fig. 5. SHA1 Hamming distance test.

make SHA1 hashes at least 50% different, so the Hamming distance test has been passed.
Results of the experiment for the SHA-512 hashing function are presented in Fig. 6 and Table 2.

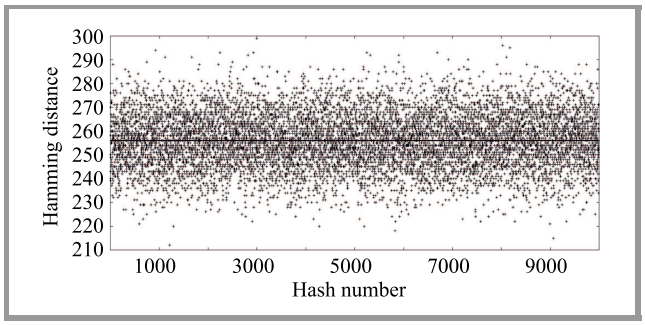

Fig. 6. SHA-512 Hamming distance test.

Table 2
Comparison of Hamming distance critical values

	Hamming distance [% values]			
Function	Max	Min	AVG	SD
SHA-1	66.25	33.75	50.04	±4.02
SHA-512	58.40	41.41	50.00	±2.21
SHA3-512	59.38	41.99	50.00	±2.22

The black line is equal to 256, because it is the expected value (distance). In 4836 out of 10,000 cases, the difference between hashes was lower than 50%, but in some case it was also close to 50%. The average distance between hash and hash′ is equal to 50% and the $|Z|$ value

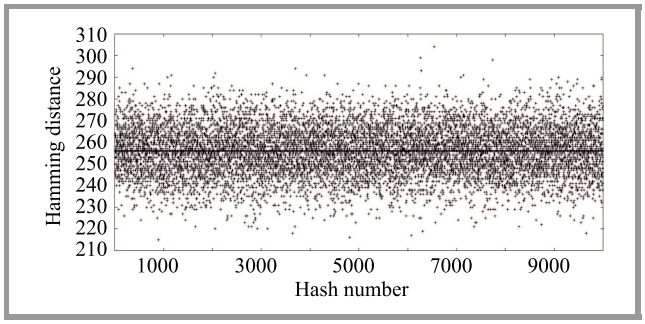

Fig. 7. SHA3-512 Hamming distance test.

was 0.148. T-Student test was also passed, but the score achieved was much better than in the SHA1 case. SHA-512 has also passed the Hamming distance test.
The research results for SHA3-512 hashing function are presented in Fig. 7 and in Table 2.
The black line is equal to 256, because it is the expected value. In 4799 out of 10,000 cases, the difference between hashes was lower than 50%. The critical values are very similar to SHA-512 (average, standard deviation). $|Z|$ statistic was equal to 0.44, thus SHA3-512 has also passed the Hamming distance test.
All three hashing functions passed the Hamming distance test, however, statistically, the SHA-512 is the best, SHA3-512 ranks second and SHA1 ranks third.

5.2. Bits Probability Test

This time, the aim was to check whether bits in the digest may be predicted or not. To measure it, we had to estimate the probability of 1s in every bit position. The ideal situation is when every bit has a 50% probability of being a 1, and a 50% probability of being a 0:

$$P_1(i) = 50\%, \;\; i = 1, \ldots, l \;, \tag{14}$$

where i denotes the bit position and l is the hash length. For each hashing function, we used 10,000 generated digests to estimate the probability of '1':

$$P_1(i) = \frac{\sum_{j=1}^{10000} hashes[j][i]}{10000}, \;\; i = 1, \ldots, l \;, \tag{15}$$

where *hashes* is a table of generated digests, j denotes j-th hash from *hashes*. We used Eq. (13) to calculate the $|Z|$ statistic. The test is passed when $|Z| < 1.96$ (significance level of $\alpha = 5\%$). The expected value is 50%.

Results of the experiment for the SHA1 hashing function are presented in Fig. (8) and in Table 3.

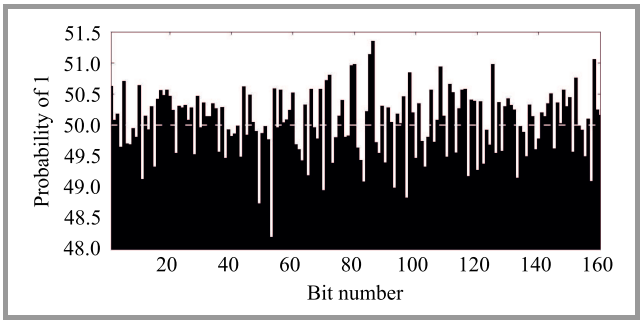

Fig. 8. SHA1 bits prediction.

As one can see, the average value is very close to 50% and the standard deviation is low. Despite the fact that none of the bits have the probability that is equal to 50%, the fluctuations are very small. The $|Z|$ statistic is equal to 1.04, so the conclusion is that none of 160 SHA1 bits can be predicted.

Table 3
Comparison of bits prediction test values

	Probability of 1			
Function	Max	Min	AVG	SD
SHA1	51.35	48.20	50.04	±0.52
SHA-512	51.46	48.49	49.99	±0.53
SHA3-512	51.74	48.76	49.99	±0.51

Results of the test for the SHA-512 hashing function are presented in Table 3 and in Fig. 9.

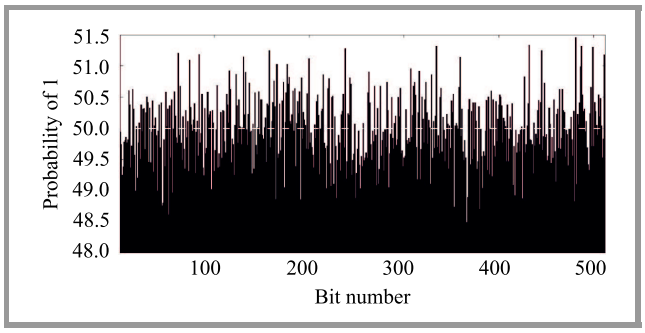

Fig. 9. SHA-512 bits prediction.

Performance of SHA-512 is a similar to that of SHA1 in the context of bits prediction. 510 of the bits have a $P_1(i)$ value that is different than 50%. However, despite small fluctuations, the average value and $|Z|$ equal to 0.863 clearly prove that none of the 512 bits of SHA-512 can be predicted.

Results of the experiment results for the SHA3-512 hashing function are presented in Table 3 and in Fig. 10.

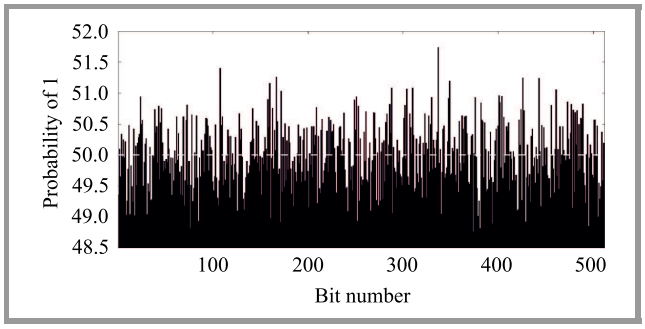

Fig. 10. SHA3-512 bits prediction.

The results for SHA3-512 are similar to those for SHA-512. 506 out of 512 bits fail to satisfy $P_1(i) = 50\%$, but the differences are small. $|Z|$ is equal to 0.317, so the conclusion is that none of 512 bits of the SHA3-512 hashing function may be predicted.

All tested functions pass the bits probability test. This test shows that in every bit position its value is random (ones and zeroes are equally probable). The best score was achieved by SHA3-512, SHA-512 ranked second and SHA1 third.

5.3. Series Test

This test allows to measure whether all hashes were generated in a random manner. Previously, in the bits prediction test, we considered each bit of digest separately, but in the context of all generated hashes. This time the existence of internal dependencies of each hash of each chosen hashing function was measured. To do this, we performed the Wald-Wolfowitz series test [21].

This measure is the subsequence taken from a sequence consisting of the same values only (0 or 1). The number of all series found in one hash will be further denoted by R. n_1 is the number of subsequences consisting only of 1s, and n_0 is the number of subsequences consisting only of 0s. For example, in the 00101101 sequence, the parameters are: $R = 6$, $n_1 = 3$ and $n_0 = 3$. The null hypothesis H_0 claims that the investigated sequence (in this case digest) is random. The alternative hypothesis H_a claims that the investigated sequence was not produced in a random manner. To decide whether H_0 is true or not, a proper test statistic value has to be calculated. Because every generated hash has $n_0 > 20$ and $n_1 > 20$, test statistics tend to have normal distribution $N(0,1)$ (when H_0 is true) and will be denoted by Z. Test statistic Z for each hash was calculated from [21]:

$$Z = \frac{R - \overline{R}}{SD} \ , \tag{16}$$

where $\overline{R}$ is the expected number of all series, such as:

$$\overline{R} = \frac{2n_0n_1}{n_0 + n_1} + 1 \ , \tag{17}$$

and SD is the standard deviation:

$$SD = \sqrt{\frac{2n_0n_1(2n_0n_1 - n_0 - n_1)}{(n_0 + n_1)^2(n_0 + n_1 - 1)}} \ . \tag{18}$$

We have chosen significance level of $\alpha = 5\%$. Thus when $|Z| > Z_{0.975}$ the H_0 is true and the hash investigated was created randomly. Parameter $Z_{0.975}$ is equal to 1.96.

Results of the experiment for the SHA1 hashing function are presented in Table 4 and Fig. 11.

Table 4
Comparison of series test critical values

	Z statistic values			
Function	Max	Min	AVG	SD
SHA1	4.23	$\tilde{0}$	0.79	±0.60
SHA-512	4.04	$\tilde{0}$	0.80	±0.61
SHA3-512	4.39	$\tilde{0}$	0.79	±0.59

The black horizontal line is indicating $Z = 1.96$ ($\alpha = 5\%$). The average value and the standard deviation show that, generally, SHA1 passes the series test, but one may notice

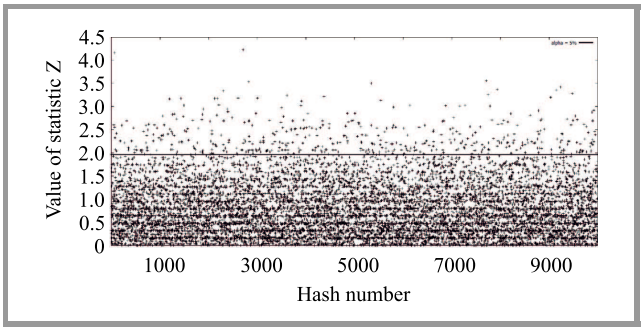

Fig. 11. SHA1 series test.

in the chart that definitely not all digests do so. 480 out of 10,000 samples (4.8%) are considered to have failed.
Results of the test for the SHA-512 hashing function are presented in Table 4 and Fig. 12.

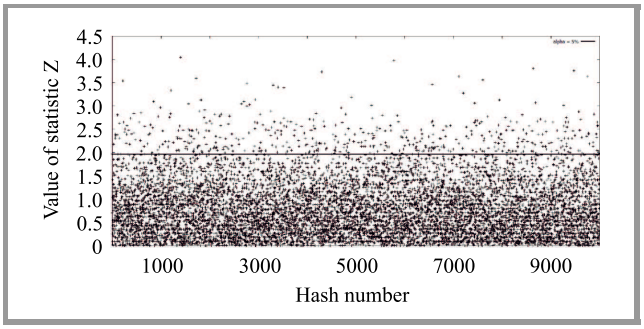

Fig. 12. SHA-512 series test.

In the case of SHA-512, the test was failed in the case of 498 out of 10,000 samples (4.98%). The value is higher than in the case of SHA1 and SHA3-512. The average Z value is the highest, however it is still far from the critical region. We can say that SHA-512 passes the series test with the worst results achieved.

Research results for the SHA3-512 hashing function are shown in Table 4 and Fig. 13.

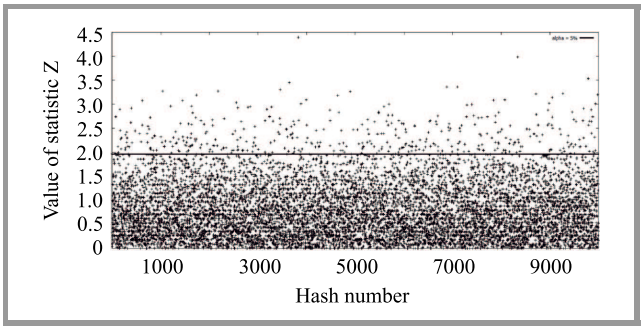

Fig. 13. SHA3-512 series test.

In contrast to SHA-512, SHA3-512 achieved the best test results. Only 417 out of 10,000 samples failed (4.17%), which is the lowest value among all hashing functions tested. The average Z value is also closest to 0. SHA3-512 definitely passes the test.

All three hashing functions have passed our last test. The best score was achieved by SHA3-512 and the worst by SHA-512. In all cases, the test was not passed by less than 5% of samples, so it may be stated that, statistically (with a significance level set to 5%), all hashes were generated randomly.

6. Summary

The aim of this paper was to describe the types, classes and main characteristics of cryptographic hash functions. The formal definition of a hash function was presented and universal hash function classes were described. Then, several methods for the construction of hash functions were disclosed. Standardization procedures for hash functions, as drawn up by NIST, USA, finalize the theoretical part of this paper.
In the research-related sections, we provided an analysis on the influence of the hashed message length on the theoretical security of hash functions, described as the number of bits of security.
The paper describes numerous experiments evaluating the basic features of SHA1, SHA-512 and SHA3-512. The randomness of such functions in terms of input spreading, single bit prediction ability and randomness inside each single bit output, were illustrated. Three tests were performed. The first was based on the Hamming distance measurement, the second examined the frequency of zeros and ones in a large sample, and the third was a series test.
Numerous experiments proved that the features of certified hash functions differ, but they all offer very good characteristics in terms of collision resistance, preimage resistance and second preimage resistance attacks.
The analysis provided may be very useful for testing new or proprietary hash functions.

References

[1] I. Gomaa, "Global information assurance certification paper", SANS Institute, May 2011.

[2] R. Sobti and G. Ganesan, "Cryptographic hash functions: A review", *Int. J. of Com. Sci. Issues*, vol. 9, no. 2, pp. 461–479, 2012.

[3] J. L. Carter and Mark N. Wegman, "Universal classes of hash functions", *J. of Comp. and Syst. Sciences*, vol. 18, no. 2, pp. 143–154, 1979 (doi:10.1016/0022-0000(79)90044-8).

[4] R. C. Merkle, "Secrecy, authentication, and public key systems", Ph.D. Thesis, Department of Electrical Engineering, Stanford University, CA, USA, 1979 [Online]. Available: http://www.merkle.com/papers/Thesis1979.pdf

[5] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications", March 1995 [Online]. Available: http://www.wisdom.weizmann.ac.il/∼naor/PAPERS/uowhf.pdf

[6] B. Preneel, "Cryptographic hash functions: An overview", in *Proc. of the 6th Int. Comp. Secur. and Virus Conf. ICSVC 1993*, Lueven, Belgium, 1993 [Online]. Available: https://www.esat.kuleuven.be/cosic/publications/article-289.pdf

[7] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "Cryptographic hash functions: A survey", Tech. Rep. 95-09, vol. 4, Department of Computer Science, University of Wollongong, 1995.

[8] S. Matyas, C. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm", *IBM Techn. Disclosure Bull.*, vol. 27, no. 10A, 1985.

[9] S. Wolfram, "Random sequence generation by cellular automata", *Adv. in Appl. Mathem.*, vol. 7, no. 2, pp. 123–169, 1986 (doi: 10.1016/0196-8858(86)90028-X).

[10] J. Daemen, R. Govaerts, and J. Vandewalle, "A framework for the design of one-way hash functions including cryptanalysis of damgård's one-way function based on a cellular automaton", in *Advances in Cryptology ASIACRYPT 91*, H. Imai, R. L. Rivest, and T. Matsumoto, Eds. *LNCS*, vol. 739, pp. 82–96. Berlin Heidelberg: Springer, 1993 (doi: 10.1007/3-540-57332-1_7).

[11] B. Preneel, "The first 30 years of cryptographic hash functions and the nist SHA-3 competition", in *Topics in Cryptology – CT-RSA 2010. The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, J. Pieprzyk, Ed. *LNCS*, vol. 5985, pp. 1–14. Berlin, Heidelberg, 2010 (doi: 10.1007/978-3-642-11925-5_1).

[12] S. Harari, "Non linear non commutative functions for data integrity", in *Advances in Cryptology*, T. Beth, N. Cot, and I. Ingemarsson, Eds. *LNCS*, vol. 209, pp. 25–32. Berlin Heidelberg: Springer, 1985 (doi: 10.1007/3-540-39757-4_4).

[13] M. S. Turan *et al.*, "NISTIR 7764: Status report on the second round of the sha-3 cryptographic hash algorithm competition", Tech. Rep., NIST, 2011 (doi: 10.6028/NIST.IR.7764).

[14] "Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family", Tech. Rep., NIST, 2007.

[15] S. J. Chang *et al.*, "NISTIR 7896: Third-round report of the SHA-3 cryptographic hash algorithm competition", Tech. Rep., NIST, 2012 (doi: 10.6028/NIST.IR.7896).

[16] Fips pub 202: M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions", Tech. Rep. no. 202, NIST, 2015 (doi: 10.6028/NIST.FIPS.202).

[17] A. K. Lenstra, "Key lengths. Contribution to the handbook of information security", 2004 [Online]. Available: http://plan9.bell-labs.co/who/akl/key_lengths.pdf

[18] Bouncy Castle library [Online]. Available: https://www.bouncycastle.org (accessed: 08.2018).

[19] Class SecureRandom [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html (accessed: 08.2018).

[20] "Hamming Distance and Error Correcting Codes", Hamming Distance [Online]. Available: http://www.oxfordmathcenter.com/drupal7/node/525 (accessed: 08.2018).

[21] Wald-Wolfowitz series test [Online]. Available: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35d.htm (accessed: 08.2018).

Jacek Tchórzewski received his B.Sc. and M.Sc. degrees in Computer Science, with distinctions, from the Cracow University of Technology, Poland, in 2016 and 2017, respectively. Currently, he is a Research and Teaching Assistant at the Cracow University of Technology and a Ph.D. student at AGH Cracow University of Science and Technology.

https://orcid.org/0000-0002-0188-4253
E-mail: jacek.tchorzewski@onet.pl
AGH University of Science and Technology
30 Mickiewicza Av.
30-059 Kraków, Poland

Cracow University of Technology
24 Warszawska St
31-155 Kraków, Poland

Agnieszka Jakóbik (Krok) received her M.Sc. in Stochastic Processes from the Jagiellonian University, Cracow, Poland and the Ph.D. degree in Neural Networks from the Tadeusz Kosciuszko Cracow University of Technology, Poland, in 2003 and 2007, respectively. From 2009 she has been an Assistant Professor at the Faculty of Physics, Mathematics and Computer Science, Tadeusz Kosciuszko Cracow University of Technology. Her main scientific and didactic interests are focused mainly on cloud computing security, artificial neural networks, genetic algorithms, and additionally on cryptography.

https://orcid.org/0000-0003-4568-3944
E-mail: agneskrok@gmail.com
Cracow University of Technology
24 Warszawska St
31-155 Kraków, Poland